The tenth in a series of tutorials for the beginner to intermediate machine language programmer

Machine Language Made BASIC
Part X: Two-Dimensional Rotation

By William P. Nee

Rotating a point around the screen is a slow but not too complicated process. It mainly involves computing new x and y locations, and machine language is great at using math for doing just that. If we avoid using ROM routines and the FP registers, this process is considerably faster.

In this article we take a point at an old x and y location and revolve it to get a new x and y location. This involves switching to the old location while the new locations are plotted.

The center of the screen (128,96) is the starting point (0,0) for all coordinates. If the angle of rotation is called A, the formulas for a new x_1 and y_1 are as follows:

$$x_1 = x.\cos(a) - y.\sin(a)$$
$$y_1 = x.\sin(a) + y.\cos(a)$$

This rotates the old x,y counterclockwise A degrees to the new x_1,y_1; new x_1,y_1 is `PSET(128+X1,96-Y1)`.

Unfortunately, using sines and cosines slows down a graphics program. The BASIC program at the end of this article gives you an idea of this problem. The more points, the longer the computation time, so we will use an alternative method.

Say our angle of rotation is roughly 7.173 degrees. The sine of 7.173 is about 1/8, and the cosine is about 127/128. Both figures can be calculated quickly by using *shift* commands.

If Register A contains the old x locations then these ML commands will divide the old location by eight and produce the same result as multiplying by the sine:

```
ASRA (Divide by 2)
ASRA (Divide by 4)
ASRA (Divide by 8)
```

This also retains the plus or minus value.

If Register B contains the old y location, then a subroutine like the following will compute the cosine:

```
TFR B,A (Save Y into Register A)
ASRB (Divide by 2)
ASRB (Divide by 4)
ASRB (Divide by 8)
ASRB (Divide by 16)
ASRB (Divide by 32)
ASRB (Divide by 64)
ASRB (Divide by 128)
PSHS B

SUBA ,S+ (the number minus 1/128 of
          the number = 127/128 of the
          number)
```

Register A will contain 127/128 times the old number (the same as the old number times the cosine). Adding and subtracting the results of both operations gives us the new x_1,y_1 locations. Since we use signed numbers, no coordinate can be greater than 127 or less than -128.

However, because we use just one byte for each coodinate, the computer continually rounds off the results and eventually produces a large error. Therefore, we store each coordinate in two bytes; the first byte is the whole number, and the second byte is the two-place decimal. We use only the first byte to `PSET` the point. This means all of our shifts are actually shifts of Register D — remember, a right shift of Register D is `ASRA, RORB`. This gives us more than enough accuracy to continue plotting without causing a rounding-off error.

There is also a new way to `PSET` a given point: by converting an x,y location to the byte containing the location and then `PSET`ting the actual bit. Let's see how this is done in `PMODE 4`.

In `PMODE 4` there are 192 rows (0 to 191) of 32 bytes each. (Location $B9 gives the bytes per line for the current `PMODE`). Multiplying the y coordinate by 32 gives you the start of the row containing the byte you want. Add to that the start of the graphics page (in Location $BA/$BB). Next, figure how far over into the row we need to go. The x location can range from 0 to 255, but since there are eight bits to a byte, divide the x location by eight. Adding this to the beginning of the row location gives us the desired byte location.

Bill Nee bucked the "snowbird" trend by retiring to Wisconsin from a banking career in Florida. He spends the long, cold winters writing programs for his CoCo.

About The One-Liner Contest . . .

THE RAINBOW's One-Liner Contest has now been expanded to include programs of either one or two lines. This means a new dimension and new opportunity for those who have "really neat" programs that simply just won't fit in one line.

Here are the guidelines: The program must work in Extended BASIC, have only one or two line numbers and be entirely self-contained — no loading other programs, no calling ROM routines, no poked-in machine language code. The program has to run when typed in directly (since that's how our readers will use it). Make sure your line, or lines, aren't packed so tightly that the program won't list completely. Finally, any instructions needed should be *very* short.

Send your entry (preferably on cassette or disk) to:

THE RAINBOW
One-Liner Contest
P.O. Box 385
Prospect, KY 40059

Let's do that for the center of the screen at 128,96 ($80,$60 in Hex). First, the desired row is Y times 32, or in Hex, $60 times $20, which is $C00. If we are in PMODE 4, without disk, graphics begin at Location $600. Adding $600 to $C00 gives us $1200. The x location is 128, and 128 divided by 8 is 16. So the desired byte is 16 (Hex 10) more bytes. Adding $1200 to $10 gives us $1210. The byte containing Location 128,96 is $1210.

Now, what bit represents the x,y location? Our x location could be 0 to 255, but there are only 8 bits (Bit 7 through Bit 0) in each byte. We need to change our x location to a number between 0 and 7. This is done by AND #7. Any number AND #7 will always be between 0 and 7. A zero means the left-most bit; a 7 means the right-most bit. We must also be sure not to erase anything already set in the byte. The OR command does this since it keeps any number already in the byte and sets only the new bits.

If our AND #7 gives us 0, we need to set the left-most bit. Do this by ORing the contents of the byte with #128 (10000000 in Base 2). This will always set the left bit (Bit 7) and keep all other bits as they are. If our AND #7 was 1, we would set the next bit over by ORing the contents of the byte with #64 (01000000 in Base 2). If AND #7 results in 7, set the right bit (Bit 0) by ORing #1 (00000001 in Base 2). The following table lists the AND #7 results and the number used to OR the byte contents:

AND #7	OR BYTE	(HEX)
0	128	#$80
1	64	#$40
2	32	#$20
3	16	#$10
4	8	#$08
5	4	#$04
6	2	#$02
7	1	#$01

These OR numbers are already stored in the computer starting at $92DD. So all we need to do is load Register A with the x location and ANDA #7; load Register y with #$92DD and load Register B with the contents of the byte (in Register x). Finally, OR Register B with the "A'th" number in the table and put the results (PSET) back into the byte. Let's follow the subroutine all the way through:

```
BYTE LDA   #$ xx    xx y coordinate
     LDB   #32      bytes per line
     MUL
     ADDD  $BA      add page start (or
                    ADDA $BA)
     TFR   D,X      byte row to Register
                    x
     LDB   #$ xx    xx x location
     LSRB
     LSRB
     LSRB           divide by 8
     ABX            add it to Register x
                    (now has the byte)
BIT  LDA   #$ xx    xx x location
     ANDA  #7       change it to 0
                    through 7
     LDY   #$92DD   OR table location
PSET LDB   ,X       get current byte
                    contents
     ORB   A,Y      OR B with 0 through
                    7th number of the
                    OR table
     STB   ,X       reload byte with
                    new contents
```

Perform this routine with the computer at $92A6 (PMODE 4/2/0) or at $92C2 (PMODE 3/1) when you execute a PSET command. Follow through these routines in *ZBug* The difference between the two routines is due to the number of bytes per line in each PMODE and because the four-color modes take two bits to set colors. Our program will not need to use locations $BE and $C0 for x and y, and since it is in PMODE 4, we will not need to scale. All of this helps the program to run more quickly.

Start off with the BASIC driver program, which PSETS a series of random dots. Modify this part any way you want — the more complex, however, the longer it will take to compute and run, and the more jumbled it will look on the screen.

The machine language program first checks all the points inside a box from screen locations 65,33 to 191,159. This ensures that no point is more than 63 bits from the center of the screen at 128,96. If a bit is set, its coordinates (x-128,y-96) are stored in a table of coordinates beginning at $5200. The coordinates are stored as two-byte numbers and as each pair is stored, the count location increases by one. Depending on how many points you set, this section may take several seconds.

The program then sets up graphics Page 5. We do not have to specify the PMODE or color set since the BASIC driver program did that for us. LOOP5 will load Stack U with a scratch-pad beginning at $7000. Then it loads Register X with the start of the coordinate table at $5200 and loads Register D with the number of dots to be set, which is also put in COUNT1. LOOP3 to GET computes all of the new x_1,y_1 rotated coordinates and puts them back as two-byte numbers. GET to FINISH restores the counter and PSETs all of the coordinates as:

```
(128-X1,96-Y1),(128-Y1,96-X1)
(128-X1,96+Y1),128-Y1,96+X1)
(128+X1,96-Y1),(128+Y1,96-X1)
(128+X1,96+Y1),(128+Y1,96+X1)
```

The video screen now allows you to see the new dots while new x_1,y_1 locations are being plotted to continue the cycle. Pressing any key breaks the program and returns you to BASIC. It's a long program, but thanks to the Color Computer's ability, it executes with amazing speed. That's 320 dot coordinates being computed and plotted with every pass! □

Listing 1: DEMO

```
10 REM DEMO PROGRAM
20 PCLEAR8:PMODE 4,1:PCLS:SCREEN
 1,1:ND=4
30 DIM X(ND),Y(ND)
40 FOR N=0 TO ND
50 X(N)=RND(63):Y(N)=RND(X(N))
60 PSET(128+X(N),96-Y(N)):NEXT
70 PMODE ,5:PCLS:GOSUB 100:SCREE
N 1
80 PMODE ,1:PCLS:GOSUB 100:SCREE
N 1
90 GOTO 70
100 FOR N=0 TO ND:A=X(N):B=Y(N)
110 X(N)=A*127/128-B/8
120 Y(N)=A/8+B*127/128
130 NEXT
140 FOR N=0 TO ND
150 PSET(128-X(N),96-Y(N)):PSET(
128-Y(N),96-X(N))
160 PSET(128-X(N),96+Y(N)):PSET(
128-Y(N),96+X(N))
170 PSET(128+X(N),96-Y(N)):PSET(
128+Y(N),96-X(N))
180 PSET(128+X(N),96+Y(N)):PSET(
128+Y(N),96+X(N))
190 NEXT:RETURN
```

Listing 2: DRIVER

```
10 REM DRIVER PROGRAM
20 PCLEAR 8:CLEAR200,&H4F00-1
30 PMODE 4,1:PCLS:SCREEN 1,1
40 FOR N=0 TO 40:'NUMBER OF DOTS
50 X=RND(63):Y=RND(X)
60 PSET(128+X,96-Y)
70 NEXT
80 EXEC &H4F00
```

The COCO hardware store

CRC COMPUTERS

DISTO **Super Controller I** $99.95

A Superb Controller. Along with the included C-DOS, plug-in three more software selectable DOSes or 2764 or 27128 EPROMs burned to your liking. The Internal M.E.B. lets you add Disto incredible Super Add-ons.

DISTO SUPER CONTROLLER II

$130. **Fantastic Super Controller II**

- **Under OS-9:**
- Buffered Read/Write sector achieved without halting the CPU.
- Continual use of keyboard even while reading or writing to disk.
- System's clock no longer looses time during Read & Write.
- NMI is blocked and transferred to IRQ in software for low CPU overhead.
- Completely Interrupt driven for fast & smooth Multi-Tasking operations.
- Drivers written by KEVIN DARLING
- Now Available at your Local Radio Shack store PN 90-2009

DISTO **3 in 1 Multi-Board Adapter** $75.00

This Muti-Board is an adapter that plugs in any Disto Super Controller, Ramdisk or MEB Adapter.
It includes a new and improved Printer Port (Centronics compatible), a faster Real Time Clock (works at 2MHz.) and a true RS-232 Serial Port (external 12 volt AC adapter required). DB25 cable included.
It fits neatly inside the metal case and is still within Tandy's power limits. It also works with or without a Multi-Pak.

DISTO **SUPER ADD-ONS**

- **Real Time Clock & Printer Interface** $34.95
- **Mini EPROM Programmer** $54.95
- **Hard Disk Interface** $40.00
- **Hard Disk Interface with RS-232** $70.00
- **Super RAM 3 ZeroK Board** $24.95
- **MEB Adapter** $35.00

DISTO **Mini Controller** $74.95 NEW

- Radio Shack/Tandy controller compatible
- Works on all COCOs with or without Multi-Pack
- 2 DOS switcher
- Accepts 24 or 28 pin EPROMs
- Low power draw and Gold plated edge connectors
- DOS Included

DISTO **RS-232 SuperPack**

- A Stand-Alone (Multi-Pak required) adapter that gives the user a true RS-232 Serial Port.
- Completely compatible with OS9's ACIA software.
- Compatible with software that requires the Tandy Deluxe RS-232 Pack.
- DB-25 cable included.

$55.00

Master Card and Visa Accepted

CRC COMPUTERS INC. **1-514-383-5293** **We accept phone orders • Call for Canadian Prices Include S&H of $4 or $8 if order exceeds $75**

10802 Lajeunesse, Montreal, Quebec, Canada H3L 2E8

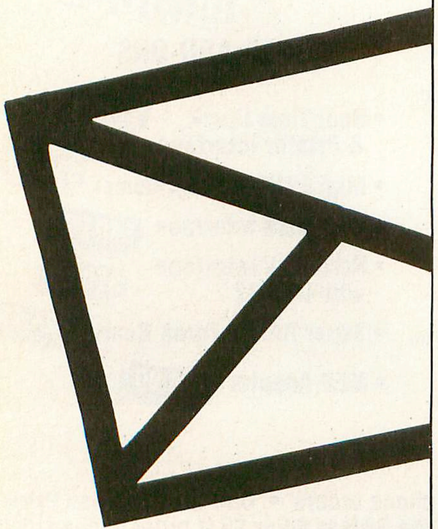
Sorry: No personal cheques

About Your Subscription

Your copy of THE RAINBOW is sent second class mail. You must notify us of a new address when you move. Notification should reach us no later than the 15th of the month prior to the month in which you change your address. Sorry, we cannot be responsible for sending another copy when you fail to notify us.

Your mailing label also shows an account number and the subscription expiration date. Please indicate this account number when renewing or corresponding with us. It will help us help you better and faster.

For Canadian and other non-U.S. subscribers, there may be a mailing address shown that is different from our editorial office address. Do not send any correspondence to that mailing address. Send it to our editorial offices at Falsoft, Inc., The Falsoft Building, P.O. Box 385, Prospect, KY 40059. This applies to everyone *except* those whose subscriptions are through our distributor in Australia.



Listing 3: ROTATION

```
4FØØ                 ØØ1ØØ         ORG    $4FØØ
4FØØ CC    52ØØ      ØØ11Ø START   LDD    #$52ØØ
4FØ3 FD    51Ø7      ØØ12Ø         STD    COORD
4FØ6 CC    ØØØØ      ØØ13Ø         LDD    #Ø
4FØ9 FD    51Ø9      ØØ14Ø         STD    COUNT
4FØC 86    21        ØØ15Ø         LDA    #33       Y START
4FØE 97    CØ        ØØ16Ø LOOP2   STA    $CØ
4F1Ø C6    41        ØØ17Ø         LDB    #65       X START
4F12 D7    BE        ØØ18Ø LOOP1   STB    $BE
4F14 BD    933C      ØØ19Ø         JSR    $933C     PPOINT(X,Y)
4F17 BD    B3ED      ØØ2ØØ         JSR    $B3ED     RESULTS TO REGISTER D
4F1A 5D              ØØ21Ø         TSTB             IS THE POINT SET?
4F1B 27    1D        ØØ22Ø         BEQ    CONT
4F1D 96    BE        ØØ23Ø STORE   LDA    $BE
4F1F 8Ø    8Ø        ØØ24Ø         SUBA   #128      X DISTANCE FROM SCREEN CENTER
4F21 5F              ØØ25Ø         CLRB
4F22 BE    51Ø7      ØØ26Ø         LDX    COORD
4F25 ED    81        ØØ27Ø         STD    ,X++      STORE 2-BYTE X COORDINATE
4F27 86    6Ø        ØØ28Ø         LDA    #96
4F29 9Ø    CØ        ØØ29Ø         SUBA   $CØ       Y DISTANCE FROM SCREEN CENTER
4F2B 5F              ØØ3ØØ         CLRB
4F2C ED    81        ØØ31Ø         STD    ,X++      STORE 2-BYTE Y COORDINATE
4F2E BF    51Ø7      ØØ32Ø         STX    COORD
4F31 FC    51Ø9      ØØ33Ø         LDD    COUNT
4F34 C3    ØØØ1      ØØ34Ø         ADDD   #1        ONE MORE POINT
4F37 FD    51Ø9      ØØ35Ø         STD    COUNT
4F3A D6    BE        ØØ36Ø CONT    LDB    $BE
4F3C 5C              ØØ37Ø         INCB
4F3D C1    BF        ØØ38Ø         CMPB   #191      REACHED MAXIMUM X COORDINATE?
4F3F 23    D1        ØØ39Ø         BLS    LOOP1
4F41 96    CØ        ØØ4ØØ         LDA    $CØ
4F43 4C              ØØ41Ø         INCA
4F44 81    9F        ØØ42Ø         CMPA   #159      REACHED MAXIMUM Y COORDINATE?
4F46 23    C6        ØØ43Ø         BLS    LOOP2
4F48 C6    Ø5        ØØ44Ø PAGE5   LDB    #5
4F4A BD    9653      ØØ45Ø         JSR    $9653
4F4D BD    9542      ØØ46Ø         JSR    $9542     PCLS
4F5Ø 8D    1B        ØØ47Ø         BSR    LOOP5
4F52 C6    Ø1        ØØ48Ø         LDB    #1
4F54 BD    95AA      ØØ49Ø         JSR    $95AA
4F57 C6    Ø1        ØØ5ØØ PAGE1   LDB    #1
4F59 BD    9653      ØØ51Ø         JSR    $9653
4F5C BD    9542      ØØ52Ø         JSR    $9542     PCLS
4F5F 8D    ØC        ØØ53Ø         BSR    LOOP5
4F61 C6    Ø1        ØØ54Ø         LDB    #1
4F63 BD    95AA      ØØ55Ø         JSR    $95AA
4F66 AD    9F AØØØ   ØØ56Ø         JSR    [$AØØØ]   ANY INPUT?
4F6A 27    DC        ØØ57Ø         BEQ    PAGE5     IF NOT, REPEAT
4F6C 39              ØØ58Ø         RTS              END OF THE PROGRAM
4F6D CE    7ØØØ      ØØ59Ø LOOP5   LDU    #$7ØØØ    START OF "SCRATCH PAD"
4F7Ø 8E    52ØØ      ØØ6ØØ         LDX    #$52ØØ
4F73 FC    51Ø9      ØØ61Ø         LDD    COUNT
4F76 FD    51ØB      ØØ62Ø LOOP3   STD    COUNT1    TEMPORARY COUNTER
4F79 EC    84        ØØ63Ø         LDD    ,X        OLD 2-BYTE X COORDINATE
4F7B ED    C4        ØØ64Ø         STD    ,U
4F7D ED    44        ØØ65Ø         STD    4,U
4F7F 47              ØØ66Ø         ASRA             REGISTER D / 128
4F8Ø 56              ØØ67Ø         RORB
4F81 47              ØØ68Ø         ASRA
4F82 56              ØØ69Ø         RORB
4F83 47              ØØ7ØØ         ASRA
4F84 56              ØØ71Ø         RORB
4F85 47              ØØ72Ø         ASRA
4F86 56              ØØ73Ø         RORB
4F87 47              ØØ74Ø         ASRA
4F88 56              ØØ75Ø         RORB
4F89 47              ØØ76Ø         ASRA
4F8A 56              ØØ77Ø         RORB
4F8B 47              ØØ78Ø         ASRA
4F8C 56              ØØ79Ø         RORB
4F8D ED    42        ØØ8ØØ         STD    2,U
4F8F EC    C4        ØØ81Ø         LDD    ,U
4F91 A3    42        ØØ82Ø         SUBD   2,U
4F93 ED    C4        ØØ83Ø         STD    ,U
4F95 EC    Ø2        ØØ84Ø         LDD    2,X       OLD 2-BYTE Y COORDINATE
4F97 47              ØØ85Ø         ASRA             REGISTER D / 8
4F98 56              ØØ86Ø         RORB
4F99 47              ØØ87Ø         ASRA
4F9A 56              ØØ88Ø         RORB
4F9B 47              ØØ89Ø         ASRA
4F9C 56              ØØ9ØØ         RORB
4F9D ED    42        ØØ91Ø         STD    2,U
```

```
4F9F EC    C4       ØØ92Ø         LDD     ,U
4FA1 A3    42       ØØ93Ø         SUBD    2,U
4FA3 ED    81       ØØ94Ø         STD     ,X++     ROTATED 2-BYTE X COORDINATE
4FA5 EC    84       ØØ95Ø NEWY    LDD     ,X       OLD 2-BYTE Y COORDINATE
4FA7 ED    46       ØØ96Ø         STD     6,U
4FA9 ED    48       ØØ97Ø         STD     8,U
4FAB EC    44       ØØ98Ø         LDD     4,U
4FAD 47             ØØ99Ø         ASRA             REGISTER D / 8
4FAE 56             Ø1ØØØ         RORB
4FAF 47             Ø1Ø1Ø         ASRA
4FBØ 56             Ø1Ø2Ø         RORB
4FB1 47             Ø1Ø3Ø         ASRA
4FB2 56             Ø1Ø4Ø         RORB
4FB3 ED    44       Ø1Ø5Ø         STD     4,U
4FB5 EC    48       Ø1Ø6Ø         LDD     8,U      OLD 2-BYTE Y COORDINATE
4FB7 47             Ø1Ø7Ø         ASRA             REGISTER D / 128
4FB8 56             Ø1Ø8Ø         RORB
4FB9 47             Ø1Ø9Ø         ASRA
4FBA 56             Ø11ØØ         RORB
4FBB 47             Ø111Ø         ASRA
4FBC 56             Ø112Ø         RORB
4FBD 47             Ø113Ø         ASRA
4FBE 56             Ø114Ø         RORB
4FBF 47             Ø115Ø         ASRA
4FCØ 56             Ø116Ø         RORB
4FC1 47             Ø117Ø         ASRA
4FC2 56             Ø118Ø         RORB
4FC3 47             Ø119Ø         ASRA
4FC4 56             Ø12ØØ         RORB
4FC5 ED    48       Ø121Ø         STD     8,U
4FC7 EC    46       Ø122Ø         LDD     6,U
4FC9 A3    48       Ø123Ø         SUBD    8,U
4FCB E3    44       Ø124Ø         ADDD    4,U
4FCD ED    81       Ø125Ø         STD     ,X++     ROTATED 2-BYTE Y COORDINATE
4FCF FC    51ØB     Ø126Ø         LDD     COUNT1
4FD2 83    ØØØ1     Ø127Ø         SUBD    #1       DECREASE TEMPORARY COUNTER
4FD5 1Ø26 FF9D      Ø128Ø         LBNE    LOOP3
4FD9 CE    52ØØ     Ø129Ø GET     LDU     #$52ØØ   COORDINATE START
4FDC FC    51Ø9     Ø13ØØ         LDD     COUNT
4FDF FD    51ØB     Ø131Ø LOOP6   STD     COUNT1   TEMPORARY COUNTER
4FE2 86    6Ø       Ø132Ø POINT1  LDA     #96
4FE4 AØ    42       Ø133Ø         SUBA    2,U      ROTATED Y COORDINATE
4FE6 C6    2Ø       Ø134Ø         LDB     #32      BYTES PER LINE
4FE8 3D             Ø135Ø         MUL
4FE9 9B    BA       Ø136Ø         ADDA    $BA      GRAPHICS START
4FEB 1F    Ø1       Ø137Ø         TFR     D,X      REGISTER D TO REGISTER X
4FED E6    C4       Ø138Ø         LDB     ,U       ROTATED X COORDINATE
4FEF CB    8Ø       Ø139Ø         ADDB    #128     ACTUAL X COORDINATE ON SCREEN
4FF1 54             Ø14ØØ         LSRB             8 BITS PER BYTE
4FF2 54             Ø141Ø         LSRB
4FF3 54             Ø142Ø         LSRB
4FF4 3A             Ø143Ø         ABX              REGISTER X=BYTE
4FF5 A6    C4       Ø144Ø BIT1    LDA     ,U       ROTATED X COORDINATE
4FF7 8B    8Ø       Ø145Ø         ADDA    #128     ACTUAL X COORDINATE ON SCREEN
4FF9 84    Ø7       Ø146Ø         ANDA    #7       CONVERT TO A NUMBER Ø - 7
4FFB 1Ø8E 92DD      Ø147Ø         LDY     #$92DD   OR TABLE LOCATION IN ROM
4FFF E6    84       Ø148Ø         LDB     ,X       GET BYTE CURRENT CONTENTS
5ØØ1 EA    A6       Ø149Ø         ORB     A,Y      OR IT WITH OR TABLE
5ØØ3 E7    84       Ø15ØØ         STB     ,X       PSET NEW BYTE CONTENTS
5ØØ5 A6    42       Ø151Ø POINT2  LDA     2,U
5ØØ7 8B    6Ø       Ø152Ø         ADDA    #96
5ØØ9 C6    2Ø       Ø153Ø         LDB     #32
5ØØB 3D             Ø154Ø         MUL
5ØØC 9B    BA       Ø155Ø         ADDA    $BA
5ØØE 1F    Ø1       Ø156Ø         TFR     D,X
5Ø1Ø E6    C4       Ø157Ø         LDB     ,U
5Ø12 CB    8Ø       Ø158Ø         ADDB    #128
5Ø14 54             Ø159Ø         LSRB
5Ø15 54             Ø16ØØ         LSRB
5Ø16 54             Ø161Ø         LSRB
5Ø17 3A             Ø162Ø         ABX
5Ø18 A6    C4       Ø163Ø BIT2    LDA     ,U
5Ø1A 8B    8Ø       Ø164Ø         ADDA    #128
5Ø1C 84    Ø7       Ø165Ø         ANDA    #7
5Ø1E 1Ø8E 92DD      Ø166Ø         LDY     #$92DD
5Ø22 E6    84       Ø167Ø         LDB     ,X
5Ø24 EA    A6       Ø168Ø         ORB     A,Y
5Ø26 E7    84       Ø169Ø         STB     ,X
5Ø28 86    6Ø       Ø17ØØ POINT3  LDA     #96
5Ø2A AØ    42       Ø171Ø         SUBA    2,U
5Ø2C C6    2Ø       Ø172Ø         LDB     #32
5Ø2E 3D             Ø173Ø         MUL
5Ø2F 9B    BA       Ø174Ø         ADDA    $BA
5Ø31 1F    Ø1       Ø175Ø         TFR     D,X
5Ø33 C6    8Ø       Ø176Ø         LDB     #128
```

WARGAME DESIGNER II

Introducing this NEW enhanced version of our most popular COCO 3 product!

Here are just a few of the new features; Choose from keyboard or joystick control. Now you can control every phase of design and play by joystick! We've added a new enhanced icon design system. Work on new icons at 5 times actual size. No more eye strain! There's a new terrain modifier menu with default values to speed up input. New menus, more visual and audio enhancements & a super fast screen loader & more!

Wargaming & game design have never been so much fun. If you haven't tried it, NOW is your chance!

WARGAME DESIGNER II

Introductory sale priced at **ONLY $25**

WGD ICON DISK #1 528 ready made, easy to use WGD II compatible unit and terrain icons. **Just $15**

WGD STAND ALONE SCENARIOS ONLY $15 each

INVASION NORTH	ATTACK ON MOSCOW
ROBOT COMMAND	DUNGEON WARRIOR
GHOST HUNTERS	ORC AMBUSH
ZULU REVENGE	DESERT RATS
ISLAND DOMINATION	FORT APACHE
TECH WARS	ROTC

GRIDIRON STRATEGY Sale price at $18
100% ML football strategy for 1 or 2 players. The first & still the best!

WEEKLY WINNER 2.0 just $15
The only lotto program we know of that has produced winning numbers. 100% ML COCO 2 & 3 disk or tape. A proven winner

CATALOG ON DISK A good investment $3
Skeptical? See before you buy. Then deduct $3.00 from your first order.

CC3FLAGS A "risky" game. only $21
Graphics oriented and definately addictive! A game of world conquest for 1 to 6 players. COCO 3 disk only.

BLACK GRID $21
An intriguing graphics puzzel for the COCO 3. Find the hidden boxes inside the black grid. 3 play modes.

MAIL MASTER Sale priced at just $10
Get your mailing lists organized. All ML

CC3CRAM Introductory sale $12.00
Stop wasting valuable disk space with COCO 3 graphic pages. Cut most files to just 4 granules! A real space saver.

★★★ APRIL SPECIAL ★★★
Order any product listed above & get the WGD stand alone game of your choice FREE
Catalog orders excluded.
★★★★★★★★★★★★★★★★★★★★
VISA & MASTERCARD accepted FREE shipping

SPORTSware
1251 S. Reynolds Road, Suite 414
Toledo, Ohio 43615
(419) 389-1515

```
5Ø35 EØ    C4           SUBB   ,U
5Ø37 54                 LSRB
5Ø38 54                 LSRB
5Ø39 54                 LSRB
5Ø3A 3A                 ABX
5Ø3B 86    8Ø   BIT3    LDA    #128
5Ø3D AØ    C4           SUBA   ,U
5Ø3F 84    Ø7           ANDA   #7
5Ø41 1Ø8E  92DD         LDY    #$92DD
5Ø45 E6    84           LDB    ,X
5Ø47 EA    A6           ORB    A,Y
5Ø49 E7    84           STB    ,X
5Ø4B A6    42   POINT4  LDA    2,U
5Ø4D 8B    6Ø           ADDA   #96
5Ø4F C6    2Ø           LDB    #32
5Ø51 3D                 MUL
5Ø52 9B    BA           ADDA   $BA
5Ø54 1F    Ø1           TFR    D,X
5Ø56 C6    8Ø           LDB    #128
5Ø58 EØ    C4           SUBB   ,U
5Ø5A 54                 LSRB
5Ø5B 54                 LSRB
5Ø5C 54                 LSRB
5Ø5D 3A                 ABX
5Ø5E 86    8Ø   BIT4    LDA    #128
5Ø6Ø AØ    C4           SUBA   ,U
5Ø62 84    Ø7           ANDA   #7
5Ø64 1Ø8E  92DD         LDY    #$92DD
5Ø68 E6    84           LDB    ,X
5Ø6A EA    A6           ORB    A,Y
5Ø6C E7    84           STB    ,X
5Ø6E 86    6Ø   POINT5  LDA    #96
5Ø7Ø AØ    C4           SUBA   ,U
5Ø72 C6    2Ø           LDB    #32
5Ø74 3D                 MUL
5Ø75 9B    BA           ADDA   $BA
5Ø77 1F    Ø1           TFR    D,X
5Ø79 E6    42           LDB    2,U
5Ø7B CB    8Ø           ADDB   #128
5Ø7D 54                 LSRB
5Ø7E 54                 LSRB
5Ø7F 54                 LSRB
5Ø8Ø 3A                 ABX
5Ø81 A6    42   BIT5    LDA    2,U
5Ø83 8B    8Ø           ADDA   #128
5Ø85 84    Ø7           ANDA   #7
5Ø87 1Ø8E  92DD         LDY    #$92DD
5Ø8B E6    84           LDB    ,X
5Ø8D EA    A6           ORB    A,Y
5Ø8F E7    84           STB    ,X
5Ø91 A6    C4   POINT6  LDA    ,U
5Ø93 8B    6Ø           ADDA   #96
5Ø95 C6    2Ø           LDB    #32
5Ø97 3D                 MUL
5Ø98 9B    BA           ADDA   $BA
5Ø9A 1F    Ø1           TFR    D,X
5Ø9C E6    42           LDB    2,U
5Ø9E CB    8Ø           ADDB   #128
5ØAØ 54                 LSRB
5ØA1 54                 LSRB
5ØA2 54                 LSRB
5ØA3 3A                 ABX
5ØA4 A6    42   BIT6    LDA    2,U
5ØA6 8B    8Ø           ADDA   #128
5ØA8 84    Ø7           ANDA   #7
5ØAA 1Ø8E  92DD         LDY    #$92DD
5ØAE E6    84           LDB    ,X
5ØBØ EA    A6           ORB    A,Y
5ØB2 E7    84           STB    ,X
5ØB4 86    6Ø   POINT7  LDA    #96
5ØB6 AØ    C4           SUBA   ,U
5ØB8 C6    2Ø           LDB    #32
5ØBA 3D                 MUL
5ØBB 9B    BA           ADDA   $BA
5ØBD 1F    Ø1           TFR    D,X
5ØBF C6    8Ø           LDB    #128
5ØC1 EØ    42           SUBB   2,U
5ØC3 54                 LSRB
5ØC4 54                 LSRB
5ØC5 54                 LSRB
5ØC6 3A                 ABX
5ØC7 86    8Ø   BIT7    LDA    #128
5ØC9 AØ    42           SUBA   2,U
5ØCB 84    Ø7           ANDA   #7
5ØCD 1Ø8E  92DD         LDY    #$92DD
5ØD1 E6    84           LDB    ,X
5ØD3 EA    A6           ORB    A,Y
5ØD5 E7    84           STB    ,X
5ØD7 A6    C4   POINT8  LDA    ,U
5ØD9 8B    6Ø           ADDA   #96
5ØDB C6    2Ø           LDB    #32
5ØDD 3D                 MUL
5ØDE 9B    BA           ADDA   $BA
5ØEØ 1F    Ø1           TFR    D,X
5ØE2 C6    8Ø           LDB    #128
5ØE4 EØ    42           SUBB   2,U
5ØE6 54                 LSRB
5ØE7 54                 LSRB
5ØE8 54                 LSRB
5ØE9 3A                 ABX
5ØEA 86    8Ø   BIT8    LDA    #128
5ØEC AØ    42           SUBA   2,U
5ØEE 84    Ø7           ANDA   #7
5ØFØ 1Ø8E  92DD         LDY    #$92DD
5ØF4 E6    84           LDB    ,X
5ØF6 EA    A6           ORB    A,Y
5ØF8 E7    84           STB    ,X
5ØFA 33    44           LEAU   4,U
5ØFC FC    51ØB FINISH  LDD    COUNT1
5ØFF 83    ØØØ1         SUBD   #1
51Ø2 1Ø26  FED9         LBNE   LOOP6     ALL DONE YET?
51Ø6 39                 RTS
51Ø7            COORD   RMB    2
51Ø9            COUNT   RMB    2
51ØB            COUNT1  RMB    2
           4FØØ         END    START

ØØØØØ TOTAL ERRORS
```

New Max-10 Fonts

Futura 24 point 2 Disks: $29.95

Century 24 Digital 24

Longhand 24 Memphis 24

Athens 18 Thames 18 Hollow 18

Ft. Worth 18 point 14 point 12 point

And 19 More! See "The Works" ad on page 19.

Note: Actual font size is 40% larger than shown here. COLORWARE

FLOPPY FILER

Creates an alphabetized 3-column cross-referenced list of files on your disks. Find the files you need, fast! (32K DECB, Printer)

MENU MAKER

Let the CoCo write the menu routine for your next program! Just enter the text for titles and selections and CoCo does the rest! (16K Disk)

Each is $8 US, $11 Canada, Postpaid

FREE joystick directory menu program with order.

Gregory Software
Box 573
Kirkland IL 60146 (815) 522-3593

VISA MasterCard